Drawbridge

Sandboxing Windows Apps at Extreme Scale

July 11, 2012

Reuben Olinsky

Azure OS Team
(formerly, MSR:Redmond OS Group)

What is Drawbridge?

- Drawbridge is a virtualization technology for hosting isolated Windows apps

What is Drawbridge?

- Drawbridge is a virtualization technology for hosting isolated Windows apps
- Drawbridge provides key benefits of VMs
  - ***secure isolation:*** *an ill-behaved app can't compromise another app or the host*
  - ***persistent compatibility:*** *evolution of the host doesn't break the app*
  - ***live migration:*** *a running app isn't tied to the host on which it started*
- …without full VM overheads

What is Drawbridge?

- Drawbridge is a virtualization technology for hosting isolated Windows apps
- Drawbridge provides key benefits of VMs
  - ***secure isolation:*** *an ill-behaved app can't compromise another app or the host*
  - ***persistent compatibility:*** *evolution of the host doesn't break the app*
  - ***live migration:*** *a running app isn't tied to the host on which it started*
- …without full VM overheads
  - hundreds of isolated apps per physical machine (vs. ~10-20 VMs per machine)
  - provisioning and starting an app takes a second (vs. minutes)
  - snapshots generally < 10 MB (vs. 100s of MB, or GBs)
- …running unmodified Windows apps

What is Drawbridge?

- Drawbridge is a virtualization technology for hosting isolated Windows apps
- Drawbridge provides key benefits of VMs
  - ***secure isolation:*** *an ill-behaved app can't compromise another app or the host*
  - ***persistent compatibility:*** *evolution of the host doesn't break the app*
  - ***live migration:*** *a running app isn't tied to the host on which it started*
- …without full VM overheads
  - hundreds of isolated apps per physical machine (vs. ~10-20 VMs per machine)
  - provisioning and starting an app takes a second (vs. minutes)
  - snapshots generally < 10 MB (vs. 100s of MB, or GBs)
- …running unmodified Windows apps
  - unmanaged or managed
  - trusted or untrusted

Demo!

High-density server hosting: IIS workers

Who (and where) are we?

- Feature team created in Azure OS org in March
  - Now have 3.5 devs, 1 sdet, 1 pm
  - We're shipping this fiscal year!
  - We're hiring!
- Started as a research project in MSR's OS group
  - Began as an intern project in FY10
  - Ongoing research and prototyping continuing

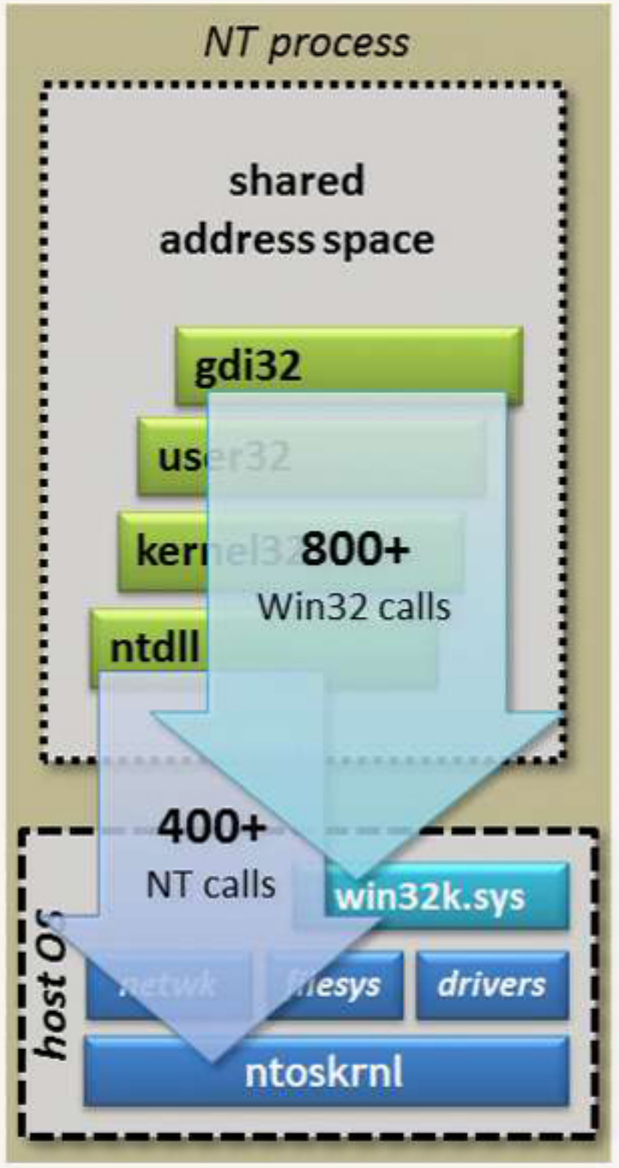
Approach

Hypervisor-based VMs for isolation?

- Yes, hardware VMs come with significant overheads
  - ***resource*** overheads: disk (GBs) and physical memory (512 MB+) per VM
  - ***runtime*** overheads: CPU (VT) virtualization, I/O virtualization, etc.
  - ***administrative*** overheads: one new OS to manage per VM
  - ***ingress/egress*** overheads: moving large VHDs to and from the cloud
- …but they offer great benefits!
  - Securely isolate guest from host
  - Support live migration
  - Only isolation mechanism strong enough to enable the cloud
- Can we retain their benefits with less overhead?
  - Most apps don't need to see virtualized hardware
  - Most apps don't require their own OS + drivers

Processes for isolation?

- NT processes are lightweight
  - Isolate address spaces
  - Scale to thousands
  - Can be organized by login sessions
  - Been around since the "dawn of time"
- …but not strong enough for cloud hosting
  - Interface with OS is huge and extensible: *1200+ functions + IOCTLs*
  - Process state distributed across kernel, drivers, services
  - NT process model was architected for *"friendly multi-tenancy"*
- ***Cloud threat model***: *anonymous hackers with unlimited access run any code of their choosing on your systems, alongside your most valued customers*
- Can we reuse processes, but with a stronger isolation interface?

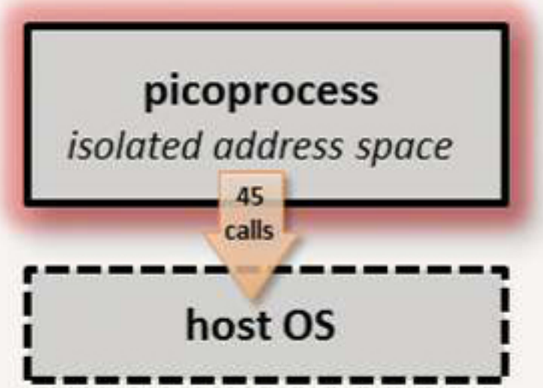


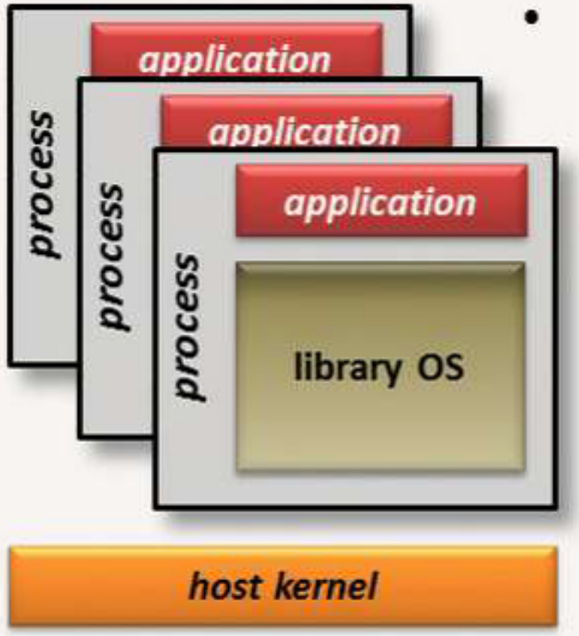
What's the Drawbridge approach?

- Key design philosophy:
  - Start with a tight, secure isolation boundary
  - Add app compatibility ***inside*** isolation container
  - ***Not*** plugging holes in a leaky but compatible interface
- Key components:
  - The *picoprocess*, an isolation mechanism
  - The *library OS*, a compatibility mechanism

Picoprocesses and library OSes

- ***Picoprocess*:** concept introduced by MSR's Xax project *(Douceur et al., 2008)*
  - Isolated address space with a *very small*, fixed interface with its host
  - Lightweight, **secure isolation container**



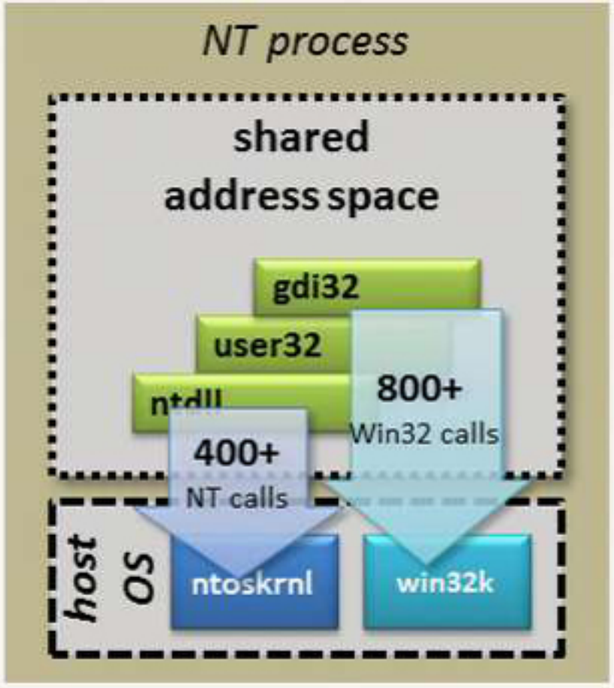


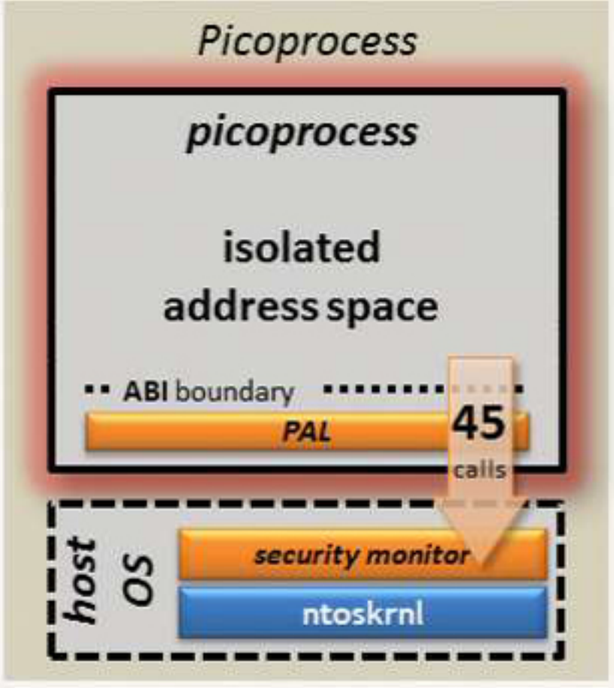
- ***Library OS***: concept championed in CS community in the '90s *(Engler et al., 1995)*
  - Minimal, shared kernel runs in supervisor mode
    - Multiplexes and abstracts hardware resources
    - Enforces cross-application protection
  - Per-app library OS **runs in user mode**
    - Constitutes OS "personality"
    - Provides application services and APIs to application
    - Runs in application's address space (user mode)
    - Each app can choose its own library OS

Implementation

The Drawbridge picoprocess on NT

- NT process with modified service handler
  - **All 1200+ system calls blocked** from user-mode (NTOS and win32k)
    - Enforced by 35-line change to `KiSystemServiceHandler`
    - No perf impact to other processes—leverages "slow path" used by UMS
  - **45 new system calls added** to process (*Drawbridge system calls*)
  - Even hard-coded traps can't break out





The Drawbridge ABI

- ***Drawbridge ABI***: interface between a Drawbridge picoprocess and its host
  - 45 downcalls, 3 upcalls – *everything else is off-limits*
  - Designed from scratch, but heavily inspired by NT
  - APIs have fixed, closed semantics (no IOCTLs)
- Analogous to VM host/guest interface, but with higher-level abstractions
  - **threads** (not virtual CPUs)
  - **virtual memory** (not physical memory)
  - **I/O streams** (not virtual device hardware)
- Design benefits:
  - ***security*** - interface is small enough to undergo manual review / inspection
  - ***portability*** - Windows apps run unmodified on any system that implements 45 functions
  - ***flexibility*** – interface allows app's state to live (almost) entirely in process

Drawbridge ABI
(excerpt)

Threading
```
DkThreadCreate
DkSemaphoreCreate
DkSemaphorePeek
DkSemaphoreRelease
DkObjectsWaitAny
...
```

Memory management
```
DkVirtualMemoryAllocate
DkVirtualMemoryFree
DkVirtualMemoryProtect
```

I/O streams
```
DkStreamOpen
DkStreamRead
DkStreamWrite
DkStreamMap
DkStreamFlush
...
```

Upcalls
```
LibOsInitialize
LibOsThreadStart
LibOsExceptionDispatch
```

Demonstrating portability

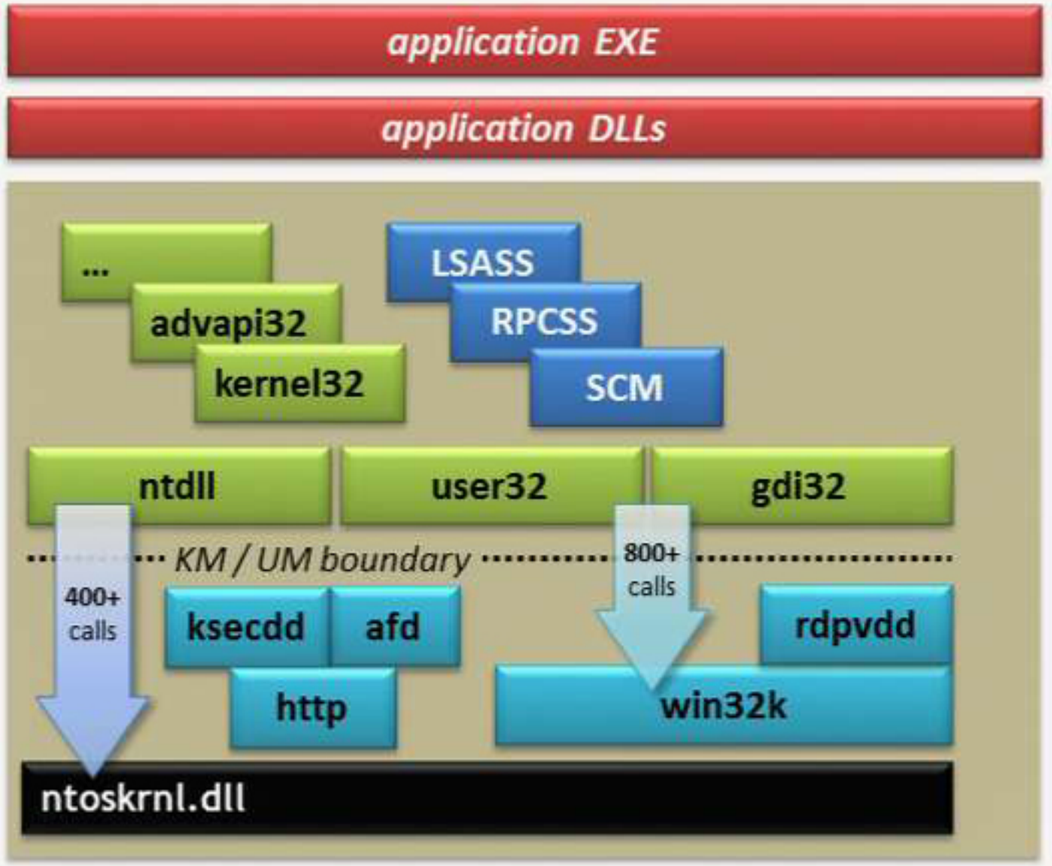
- Existing **library OS** implementations:

- Stable **host** implementations:

- Prototype **host** implementations *(past, present, and in-progress)*:

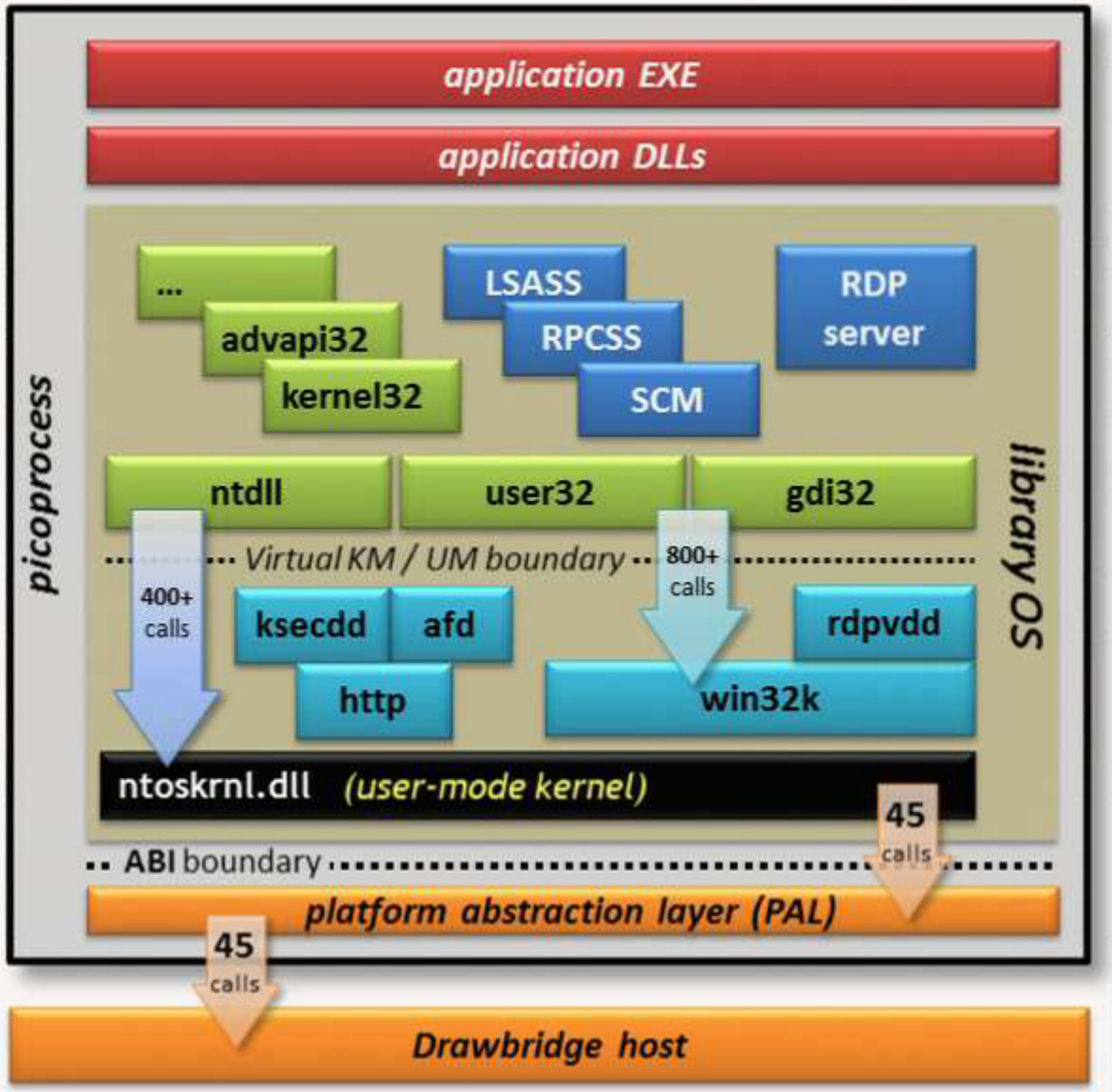
The Windows library OS

- Based on Windows OS
  - Same binaries *(where possible)*
  - Same architecture



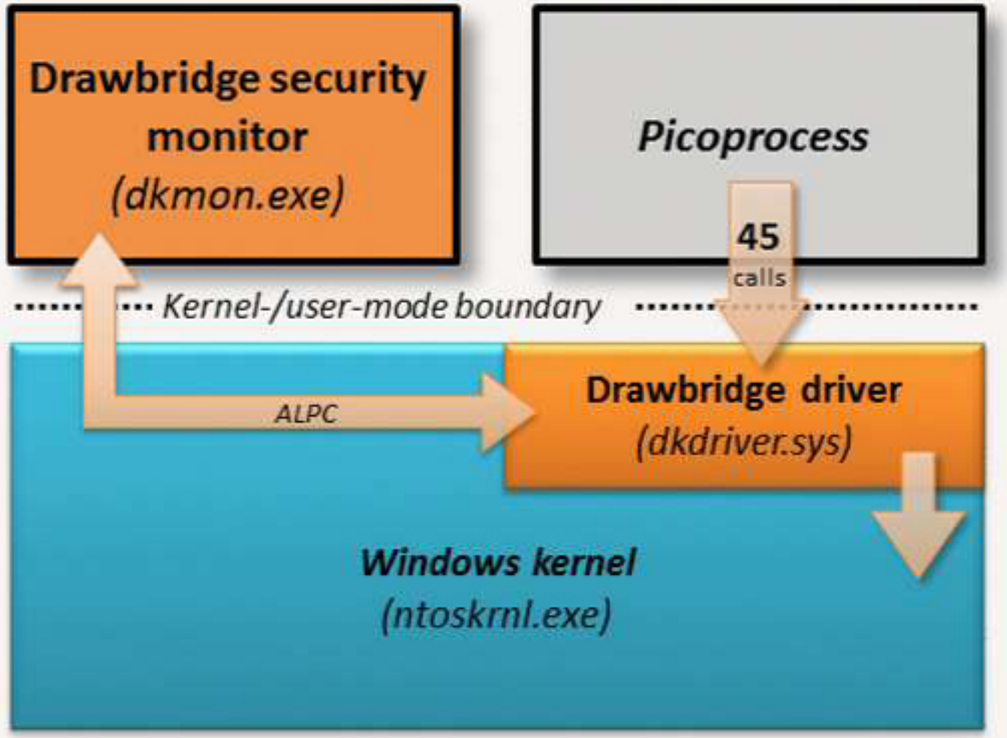
The Windows library OS

- Based on Windows OS
  - Same binaries *(where possible)*
  - Same architecture
- Windows ***enlightened*** **to run in a picoprocess** with the app
  - lifted into user mode
  - most changes in user-mode kernel
- Example library OS: *Win7 SP1*
  - **100MB on disk** (~150 DLLs)
  - **16MB of working set** + app
  - 5.5+ MLoC for 15,000+ Win32 APIs
- Each picoprocess runs its own library OS
  - app chooses its library OS
  - version need not match across picoprocesses or host



The Drawbridge-on-Windows host

- ***Drawbridge host*** implements 45-function ABI atop Windows
- Analogous to Hyper-V's hypervisor + virtualization stack
- Split between kernel-mode driver and user-mode worker
  - Driver implements ABI
  - Driver consults security monitor for policy decisions



The Drawbridge security monitor

- ***Security monitor*** – user-mode half of Drawbridge host
  - launches app in picoprocess
  - makes access policy decisions
  - "normal" NT process

The Drawbridge security monitor

- ***Security monitor*** – user-mode half of Drawbridge host
  - launches app in picoprocess
  - makes access policy decisions
  - "normal" NT process
- Policy decisions based on *manifests*
  - All external resources are blocked by default
  - Resources can be white-listed back in by admin
  - Access specified via virtual to physical namespace mappings

Sample Policy

```
[Namespace.Writable]
pipe.server:///RDP=pipe.server:///RDP_Drawbridge          ; expose 'RDP' named pipe server out of
                                                          ; process as 'RDP_Drawbridge'
tcp.server://localhost:3000=tcp.server://localhost:3000   ; allow app to listen (only) on port 3000
tcp.client:=tcp.client:                                   ; allow use of any TCP client socket

[Namespace]
file:///users/jdoe/documents=file:///documents            ; allow R/O access to Documents folder
```

Drawbridge packages

- ***Drawbridge package*** – self-contained, self-describing unit of deployment
- A package contains:
  - Manifest
    - Identity (name, version, options)
    - Dependencies on other packages
    - Access control policy requirements
    - Relative paths to important contained files (e.g. app EXE)
  - Files
  - Registry data (.reg format)
  - Debug resources (e.g. symbols, etc.)
- Everything's a package: app, library, library OS, suspended app
- Security monitor resolves transitive closure of packages and dependencies
  - **File content from packages is unioned** into virtual FS
  - **Registry content from packages is unioned** into virtual registry
  - Packages are read-only, mapped copy-on-write

Sample Manifest

```
[Package]
ManifestVersion=1
PackageRevision=4

[Identity]
Name=IISWorker
MajorVersion=7
MinorVersion=5
BuildNumber=7601
Architecture=x64

[Dependency.Win7]
Name=Windows
MajorVersion=6
MinorVersion=1

[Dependency.CLR4]
Name=MicrosoftNET
MajorVersion=4
MinorVersion=0

[Windows.Application]
Exe=package:///windows/system32/inetsrv/w3wp.exe

[Windows.Registry]
File:///w3wp.exe.dbreg
```

Demos!

- Peeking inside a picoprocess: **PowerShell + .NET 4**
- Suspending and resuming apps
- Launching multiple apps

So… what's next?

- ***Ship it! … in Azure … before the end of CY12***
  - Resource metering / performance isolation
  - Performance work
  - Improve app compat
  - Fix bugs
- Prep for integration into Windows v.Next to support Azure scenarios
- In-parallel research continuing in MSR (in private branches)
  - CloudTop (Brian Meyers)
  - TCP/IP stack in library OS (Brian Zill)
  - Non-Windows library OS (Andrew Baumann)
  - Incremental checkpointing (Jay Lorch)
  - DirectX support (MSR-A)

Learn more about Drawbridge!

- Visit our portal (includes pointers to code base, builds):
  http://drawbridge
- Contact us:
  mailto:AzDraw

Questions?